MA4832 Microprocessor Systems
Lab Exercise 4 – Programming PWM

In this session, you will learn how to

- Use Timing system to generate Pulse Width Modulation (PWM) signals to control a DC motor and Square waves to blink an onboard LED
- Display the generated waveforms on an oscilloscope
- Adjust the pulse width to vary the speed of the motor
- Adjust the frequency to vary the LED blinking speed

1. Generation of PWM signals using PWM modules

- Schematic of the sample circuit is shown in Fig. 1.
- Programs:
  1) Prog_PWM.s (main program)
  2) Prog_PLL.s (configure system clock)
  3) Startup.s

```
; Prog_PWM.s
;
; PWM Output on PB6 (M0PWM0) pin
;

               IMPORT PLL_Init       ; symbol names defined in other file, named Prog_PLL.s


GPIO_PORTB_AFSEL_R     EQU 0x40005420
GPIO_PORTB_DEN_R       EQU 0x4000551C
GPIO_PORTB_AMSEL_R     EQU 0x40005528
GPIO_PORTB_PCTL_R      EQU 0x4000552C        ;p688

SYSCTL_RCGCGPIO_R      EQU 0x400FE608        ; GPIO Run Mode Clock Gating Control
SYSCTL_RCGCPWM_R       EQU 0x400FE640        ; PWM Run Mode Clock Gating Control

PWM0_CTL_R             EQU 0x40028040        ; PWM0 Control
PWM0_GENA_R            EQU 0x40028060        ; PWM0 Generator A Control
PWM0_CMPA_R            EQU 0x40028058        ; PWM0 Compare A
PWM0_LOAD_R            EQU 0x40028050        ; PWM0 Load
PWM0_ENABLE_R          EQU 0x40028008        ; PWM0 Output Enable

SYSCTL_RCC_R           EQU 0x400FE060        ; Run-Mode Clock Configuration p254


               AREA    |.text|, CODE, READONLY, ALIGN=2
               THUMB
               EXPORT  Start

Start
               BL PLL_Init    ; call subroutine (in Prog_PLL.s) to generate system clock of
40MHz

; initialise PortB and PWM Module
;
; activate clock for Port B
               LDR R1, =SYSCTL_RCGCGPIO_R    ;
               LDR R0, [R1]                  ;
               ORR R0, R0, #0x02             ; turn on clock for GPIOB
               STR R0, [R1]
               NOP                           ; allow time for clock to finish
               NOP
               NOP
```

```
; activate clock for PWM
               LDR R1, =SYSCTL_RCGCPWM_R     ;
               LDR R0, [R1]                  ;
               ORR R0, R0, #0x01             ; turn on clock for PWM Module 0
               STR R0, [R1]
               NOP                           ; allow time for clock to finish
               NOP
               NOP

;
; disable analog functionality
               LDR R1, =GPIO_PORTB_AMSEL_R
               LDR R0, [R1]
               BIC R0, R0, #0x40             ; disable PB6 analog function
               STR R0, [R1]                  ; M0PWM0 is PB6 pg 1351

; select alternate function
               LDR R1, =GPIO_PORTB_AFSEL_R
               LDR R0, [R1]
               ORR R0, R0, #0x40             ; enable alternate function on PB6
               STR R0, [R1]

; configure as M0PWM0 output
               LDR R1, =GPIO_PORTB_PCTL_R
               LDR R0, [R1]
               BIC R0, R0, #0x0F000000
               ORR R0, R0, #0x04000000       ; assign PB6 as M0PWM0 pin
               STR R0, [R1]


; enable digital port
               LDR R1, =GPIO_PORTB_DEN_R
               LDR R0, [R1]
               ORR R0, #0x40                 ; enable digital I/O on PB6
               STR R0, [R1]

; set PWM clock of 0.625MHz
               LDR R1, =SYSCTL_RCC_R
               LDR R0, [R1]
               BIC R0, R0, #0x000E0000       ; use PWM clock divider as the source for PWM
clock
               ORR R0, R0, #0x001E0000       ; pre-divide system clock down for use as the
timing
                                             ; ref for PWM module
               STR R0, [R1]                  ; Divisor "/64" -> PWM clock = 40MHz/64 =
0.625MHz
                                             ; (clock period = 1/0.625 = 1.6 µs)

; configure PWM generator 0 block
               LDR R1, =PWM0_CTL_R           ; PWMnCTL p1266
               LDR R0, =0x00                 ; select Count-Down mode and
               STR R0, [R1]                  ; disable PWM generation block


; control the generation of pwm0A signal
               LDR R1, =PWM0_GENA_R          ; PWMnGENA P1282
               LDR R0, =0x8C                 ; pwm0A goes high when the counter matches
                                             ; comparator A while counting down
               STR R0, [R1]                  ; and drive pwmA Low when the counter matches
value
                                             ; in the PWM0LOAD register
;
; set duty cycle, about 75%
;
; comparator match value -> set PWM space time (off time)
               LDR R1, =PWM0_CMPA_R
               LDR R0, =400                  ; store value 400(decimal) into comparator A, PB6
                                             ; goes high when match
               STR R0, [R1]                  ; PWM "space" time is 0.64 ms (400 x 1.6 µs)


; counter load value -> set period
               LDR R1, =PWM0_LOAD_R          ; load value = 1600(decimal) ->
                                             ; pulse period = 1600 x 1.6 µs = 2.56 ms
               LDR R0, =1600                 ; In count-down mode, this value is loaded into
the
```

```
                                        ; counter after it reaches zero & PB6 pin goes
                                        low
                STR R0, [R1]

                        LDR R1, =PWM0_CTL_R           ;
                LDR R0, =0x01                   ; enable PWM generation block & produces PWM
signal
                STR R0, [R1]

                LDR R1, =PWM0_ENABLE_R          ; enable M0PWM0 pin
                LDR R0, =0x01                   ;
                STR R0, [R1]

Loop
                B       Loop

                ALIGN                           ; make sure the end of this section is aligned
                END                             ; end of file
```

- Load and run the program.
- Observe the following points:
  (i) The generated PWM signal at M0PWM0 pin (PB6) is displayed on the oscilloscope
  (ii) The speed of the motor can be adjusted by varying store value in Comparator A

2. Generation of Square waves (50% duty cycle) using Timer Interrupt of GPTM (General-Purpose Timer Module)
   - Schematic of the sample circuit is the same as Fig. 1.
   - Programs:
     1) Prog_Main_Periodic.s (main program)
     2) Prog_Timer_Init.s (set up Timer)
     3) Prog_PLL.s (configure system clock)
     4) Startup.s

```
; Prog_Main_Periodic.s
;
; Main Program: Generate PWM using Timer Interrupt
;

; PLL_Init and Timer_Init are symbol names defined in a separately assembled source files

               IMPORT   PLL_Init
               IMPORT   Timer_Init


TIMER0_ICR_R           EQU 0x40030024         ; GPTM Interrupt Clear
TIMER_ICR_TATOCINT     EQU 0x00000001         ; GPTM TimerA Time-Out Raw Interrupt

GPIO_PORTF2            EQU 0x40025010
GPIO_PORTF_DIR_R       EQU 0x40025400
GPIO_PORTF_AFSEL_R     EQU 0x40025420
GPIO_PORTF_DEN_R       EQU 0x4002551C
GPIO_PORTF_AMSEL_R     EQU 0x40025528
GPIO_PORTF_PCTL_R      EQU 0x4002552C
SYSCTL_RCGCGPIO_R      EQU 0x400FE608

               AREA    |.text|, CODE, READONLY, ALIGN=2
               THUMB
               EXPORT  Timer0A_Handler
               EXPORT  Start

;
; Timer0A Interrupt Handler
;
Timer0A_Handler                              ; execute every 0.25 second

; toggle PF2 LED
               LDR R1, =GPIO_PORTF2          ;
               LDR R0, [R1]                  ; read PF2
               EOR R0, R0, #0x04             ; R0 = R0^0x04 (toggle PF2)
               STR R0, [R1]                  ; store PF2

               LDR R1, =TIMER0_ICR_R         ; write "1" to clear interrupt before
               LDR R0, =0x01                 ; returning to main program
               STR R0, [R1]                  ;

               BX LR                        ; return from interrupt

Start
               BL PLL_Init           ; call subroutine (in Prog_PLL.s) to generate system
                                     ; clock of 40MHz -> period = 1/40MHz = 0.025 µs


; Initialise Port F
; configure PF2 as GPIO, digital output (disable alternate and analog functions)

; activate clock for Port F

               LDR R1, =SYSCTL_RCGCGPIO_R    ;
               LDR R0, [R1]
               ORR R0, R0, #0x20             ; turn on clock for GPIOF
               STR R0, [R1]
               NOP
```

```
                NOP                             ; allow time to finish activating

; set direction register
                LDR R1, =GPIO_PORTF_DIR_R
                LDR R0, [R1]
                ORR R0, R0, #0x04               ; set PortF bit 2 (PF2) output
                STR R0, [R1]

; regular port function
                LDR R1, =GPIO_PORTF_AFSEL_R
                LDR R0, [R1]
                BIC R0, R0, #0x04               ; R0 = R0&~0x04 (disable alternate function on
PF2)
                STR R0, [R1]

; enable digital port
                LDR R1, =GPIO_PORTF_DEN_R
                LDR R0, [R1]
                ORR R0, R0, #0x04               ; R0 = R0|0x04 (enable digital I/O on PF2)
                STR R0, [R1]

; configure as GPIO
                LDR R1, =GPIO_PORTF_PCTL_R
                LDR R0, [R1]
                BIC R0, R0, #0x00000F00         ; R0 = R0&~0x00000F00 (clear port control field
for PF2)
                ADD R0, R0, #0x00000000         ; R0 = R0+0x00000000 (configure PF2 as GPIO)
                STR R0, [R1]

; disable analog functionality
                LDR R1, =GPIO_PORTF_AMSEL_R
                MOV R0, #0                      ; disable analog function on PortF
                STR R0, [R1]

;
; enable Timer0A interrupt every 0.25 second
;
                LDR R0, =10000000               ; initialize Timer0A for 0.25 second interrupts
                                                ; (0.025 µs * 10000000 = 0.25 sec)
                BL Timer_Init                   ; call subroutine (in Prog_Timer_Init.s) for
Timer
                                                ; module 0_Timer A set up
                CPSIE I                         ; enable IRQ interrupt

loop
                WFI                             ; wait for interrupt
                B loop                          ;

                ALIGN                           ; make sure the end of this section is aligned
                END                             ; end of file
```

- Load and run the program
- Observe the following points:
  (i) The generated Square waves (at pin PF2) is displayed on the oscilloscope
  (ii) The LED blinking speed can be adjusted by varying the stored value in Register R0

Exercise:
Modify the first program, Prog_PWM.s, such that

- The opto switch (connected to PA3) is used as an input device to start the motor turning. After the motor has started turning, the microprocessor can ignore the input signals from the opto switch.

- The external rotary potentiometer connected to ADC pin AIN8 (at pin PE5) is used as a real time controller of the motor's speed. Depending on the converted 12-bit reading at PE5, three distinct speed levels are set accordingly:

PE5 Reading (Hex)	**Value of PWM Mark time (High/On time)**
000 to 400	200
401 to C00	800
C01 to FFF	1400

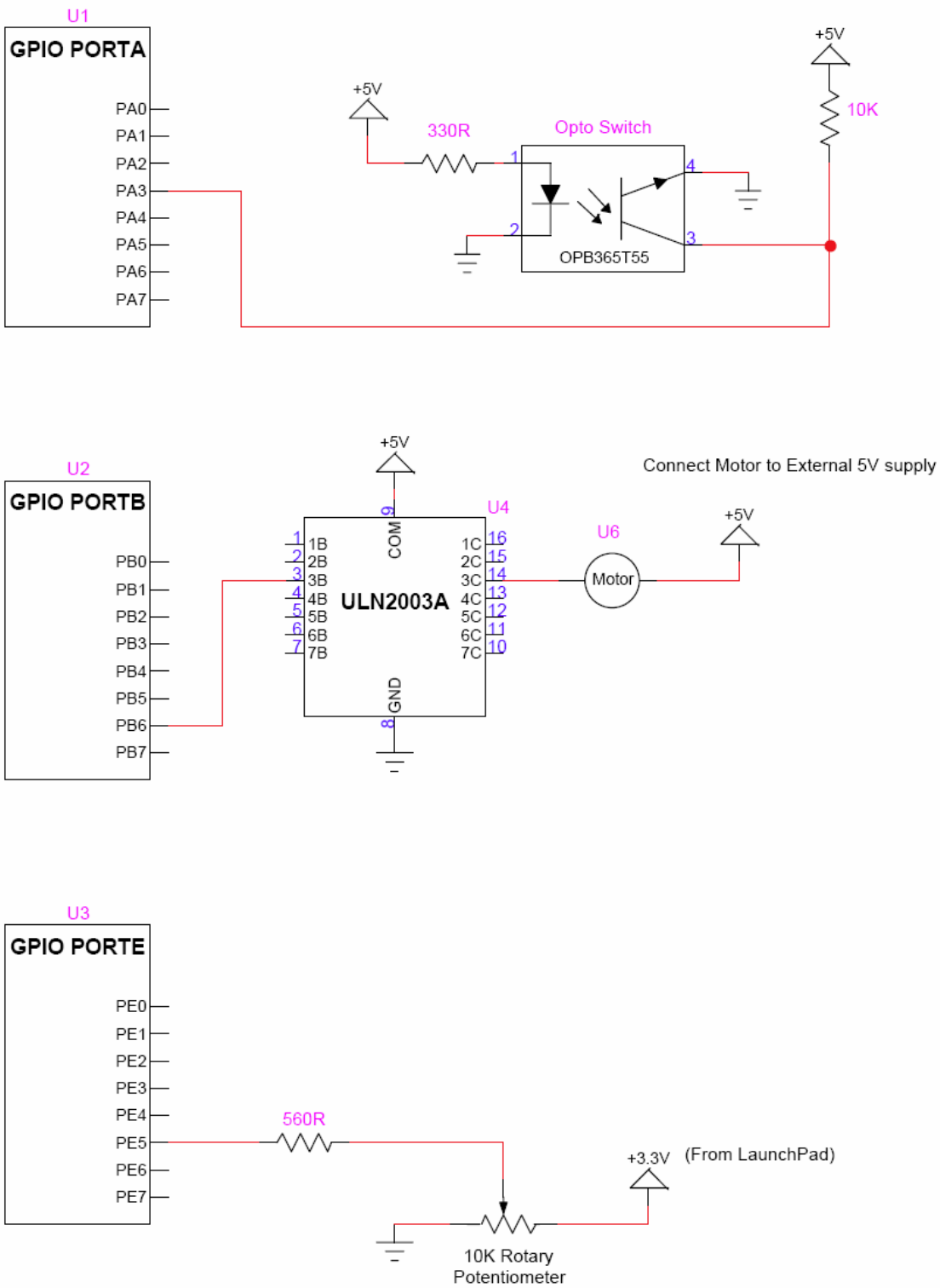

Fig. 1 Schematic Diagram